2

MASSACHUSETTS INSTITUTE OF TECHNOLOGY VLSI PUBLICATIONS

VLSI Memo No. 89-535
May 1989

Parallel Graph Contraction

Cynthia A. Phillips

Abstract

This paper shows how n-node, e-edge graphs can be *contracted* in a manner similar to the parallel tree contraction algorithm due to Miller and Reif. We give an $O((n + e)/\lg n)$-processor deterministic algorithm that contracts a graph in $O(\lg^2 n)$ time in the EREW PRAM model. We also give an $O(n/\lg n)$-processor randomized algorithm that with high probability can contract a bounded-degree graph in $O(\lg n + \lg^2 \gamma)$ time, where γ is the maximum genus of any connected component of the graph. (The algorithm can be made to run in deterministic $O(\lg n \lg^* n + \lg^2 \gamma)$ time using known techniques.) This algorithm does not require *a priori* knowledge of the genus of the graph to be contracted. The contraction algorithm for bounded-degree graphs can be used directly to solve the problem of region labeling in vision systems, i.e., determining the connected components of bounded-degree planar graphs in $O(\lg n)$ time, thus improving the best previous bound of $O(\lg^2 n)$.

Microsystems Research Center Room 39-321 | Massachusetts Institute of Technology | Cambridge Massachusetts 02139 | Telephone (617) 253-8138

Acknowledgements

To appear in *First Annual ACM Symposium on Parallel Algorithms and Architectures*, June 1989. This research was supported in part by the Defense Advanced Research Projects Agency under contract number N00014-87-K-0825, the Office of Naval Research under contract number N00014-86-K-0593, and an International Business Machines Corporation graduate fellowship.

Author Information

Phillips: Laboratory for Computer Science, Room NE43-338, MIT, Cambridge, MA, 02139. (617) 253-7583.

Parallel Graph Contraction

Cynthia A. Phillips

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract

This paper shows how n-node, e-edge graphs can be *contracted* in a manner similar to the parallel tree contraction algorithm due to Miller and Reif. We give an $O((n+e)/\lg n)$-processor deterministic algorithm that contracts a graph in $O(\lg^2 n)$ time in the EREW PRAM model. We also give an $O(n/\lg n)$-processor randomized algorithm that with high probability can contract a bounded-degree graph in $O(\lg n + \lg^2 \gamma)$ time, where γ is the maximum genus of any connected component of the graph. (The algorithm can be made to run in deterministic $O(\lg n \lg^* n + \lg^2 \gamma)$ time using known techniques.) This algorithm does not require *a priori* knowledge of the genus of the graph to be contracted. The contraction algorithm for bounded-degree graphs can be used directly to solve the problem of region labeling in vision systems, *i.e.*, determining the connected components of bounded-degree planar graphs in $O(\lg n)$ time, thus improving the best previous bound of $O(\lg^2 n)$.

1 Introduction

The parallel tree contraction technique of Miller and Reif [23] has proved to be a valuable tool in many parallel graph algorithms. This paper provides a similar contraction technique that applies not only to trees, but also to general graphs. It also provides a second, potentially faster contraction technique for bounded-degree graphs (and general graphs when an embedding is known).

A *contraction step* of an undirected graph $G = (V, E)$ with respect to an edge $(u, v) \in E$ is the operation that replaces u and v by a new vertex w which is adjacent to all those vertices to which u and v were adjacent. If vertices u and v have an adjacent vertex x in common, the contraction step produces only one edge between w and x rather than two. Thus, the graph that results from a contraction step is not a multigraph. A *parallel paired contraction step* is a vertex-independent sequence of contraction steps. A *parallel multi-contraction step* is a sequence of contraction steps with respect to an acyclic set of edges. We refer to the sequence as simply a parallel contraction step whenever the type of contraction is clear from context. A parallel paired contraction step of any bounded-degree graph can be implemented in constant parallel time. A *(parallel) contraction* of a graph is the process by which a connected graph is reduced to a single node by iterated (parallel) contraction steps.

Miller and Reif [23] show how any tree can be contracted in $O(\lg n)$ time using $O(n)$ processors in the CRCW PRAM model using randomization. (They also show how to reduce the number of processors to $O(n/\lg n)$) They give an $O(\lg n)$-time algorithm for tree contraction that is deterministic, but the deterministic algorithm does not perform contraction in the strict sense described above. Tree contraction can be made to run in randomized $O(\lg n)$ time and in deterministic $O(\lg n \lg^* n)$ time using $O(n)$ processors in the EREW and DRAM models, as was shown in [20].

squared

In this paper, we use the technique of parallel contraction in a more general setting. We show that a remarkably simple contraction algorithm that uses $(n+e)/\lg n$ processors reduces a connected n-node e-edge graph to a single node in $O(\lg^2 n)$ steps and a second simple algorithm which uses the first as a subroutine, reduces a connected bounded-degree graph to a single node ~~using $n/\lg n$ processors~~ in $O(\lg n + \lg^2 \gamma)$ steps, where γ is the maximum genus of any connected component of graph G. The second algorithm immediately yields an asymptotically efficient solution to the problem of region labeling in vision systems, as well as to solutions of other planar graph problems. (KR)

The genus of a graph is the sum of the geni of the connected components. That is, if the ith connected component has genus γ_i, then the genus of the graph is $\gamma_G = \sum_i \gamma_i$. The running time of our second algorithm, however, depends only upon the maximum genus of any connected component, (i. e. $\gamma = \max_i \gamma_i$). Henceforth, when we refer to the genus of a graph G, we will use the notation γ_G when we mean the true genus and we will use the notation γ when we mean the maximum genus of any connected component of graph G.

Figure 1 illustrates the data structure we use in our algorithms. We represent each vertex of degree d by a doubly-linked ring of d *real processors* alternating with d *dummy processors*. The dummy processors are needed for efficient implementation of a parallel multi-contraction step. The edges between real processors and dummy processors, represented by heavy lines in figure 1, are called *vertex edges*

[c] This research is supported in part by the Defense Advanced Research Projects Agency under Contract N00014-87-K-0825 and in part by the Office of Naval Research under Contract N00014-86-K-0593 and in part by an IBM graduate fellowship.

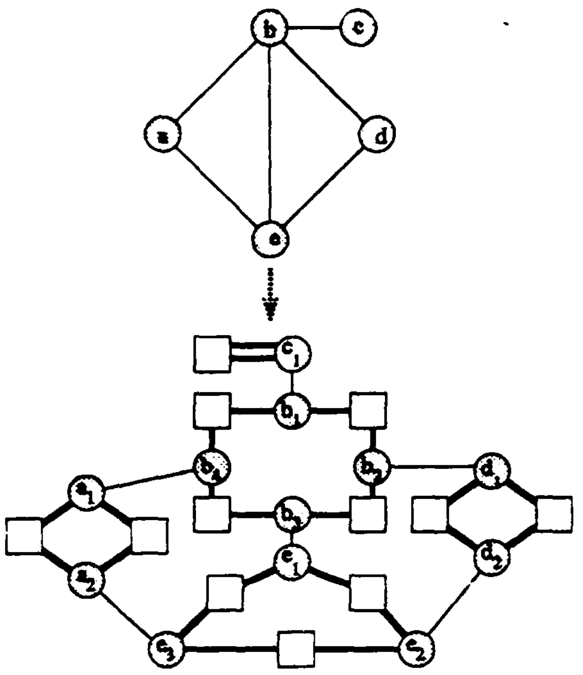

Figure 1: We represent graphs using a ring of processors for each vertex. Two (round) *real* processors are separated by a (square) *dummy* processor in the list. Vertex edges, drawn as heavy lines, go between processors in the same vertex ring, while graph edges (thin lines) go between processors in different vertex rings.

and edges between vertex rings are called *graph edges*. Each processor in a vertex ring knows the ID of the vertex, defined to be the maximum identifier of any processor in the vertex ring.

The remainder of the paper is organized as follows. Section 2 presents the contraction algorithm for general graphs, argues its correctness, and analyses its running time. Section 3 presents the contraction algorithm for bounded-degree graphs and Section 4 analyzes its running time using a "missing edge" lemma which is proved in section 5. Section 6 shows how graph contraction can be applied to various graph problems, including the region-labeling problem. Section 7 offers some concluding remarks.

2 General Algorithm

This section presents an $O(n+e)$-processor contraction algorithm for general graphs, argues its correctness and analyzes its running time. We describe how to contract an edge and we give a strategy for determining a set of acyclic edges to contract in a parallel multi-contraction step. We then discuss implementation issues such as avoiding sorting, maintaining linear space, and reducing the processor count to $O((n+e)/\lg n)$.

Figure 2 illustrates how to contract an edge (u,v) by using the dummy processors to merge the two vertex rings and splice out the real processors corresponding to the contracting edge. The dummy edges allow us to simultaneously contract any acyclic set of edges, including trees of arbitrary depth, in constant time. We then pay an extra $O(\lg n)$ time to remove the extra dummy vertices.

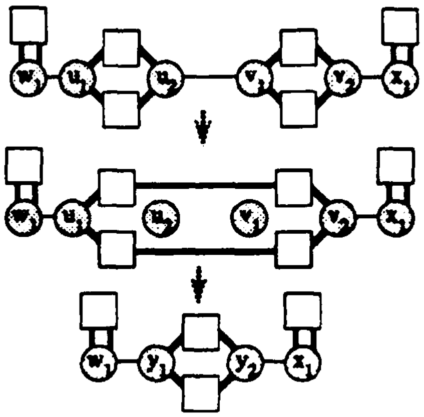

Figure 2: To contract a graph edge (u,v), we use the neighboring dummy processors on each side to merge the vertex rings and splice out the real processors. We later clean up the extra dummy processors. Dummy processors allow contraction of trees of arbitrary depth.

The algorithm for contracting graphs is simple. It uses a subroutine MERGE(u,v) which performs the edge contraction described above for edge (u,v).

```
Algorithm CONTRACT
1 While ∃ a vertex with deg ee > 0 do in parallel
2   Each vertex u with degree > 0, do in parallel
3     Let v be the neighbor of highest ID
4     MERGE(u, v)
5     Remove extra dummy processors
6     Remove multiple edges to adjacent vertices
7     Propagate new ID.
```

We now argue the correctness of Algorithm CONTRACT. The only subtle point is that simultaneous contraction of a set of edges that form a tree results in a single vertex ring. If we were to contract edges that form a cycle, then we may end up with multiple rings, thus severing a connected component. Because each vertex chooses to merge with its neighbor of maximum ID, the edges chosen to contract in an iteration cannot form a cycle.

We now analyze the running time of algorithm CONTRACT. Each iteration of the loop requires $O(\lg n)$ time. The loop termination check in line 1 can be done in $O(\lg n)$ time. Each processor can determine in constant time whether the vertex ring to which it belongs has degree > 0. We can remove multiple edges in $O(\lg n)$ time by first sorting the processors in a vertex ring with respect to the ID of the neighbor across the graph edge and then pointer jumping. Similarly, propagating the highest ID of any neighbor to all processors in a ring, removing extra dummy processors, and propagating the new vertex ID can all be done in $O(\lg n)$ time using pointer jumping. If we are careful, all pointer jumping can be done without using concurrent reads. The MERGE operation requires only constant time. Thus each iteration requires $O(\lg n)$ time. The algorithm terminates in $O(\lg n)$ iterations because each vertex merges on every iteration, at least halving the number of vertices.

We can avoid sorting if we allow multiple edges to remain after contraction and instead use pointer jumping to remove the self edges that appear in later iterations. If we choose this option, then in line 3 we must also break ties among all edges to vertex v. Using the idea of "spares" from [20], the algorithm can be made to use only linear space.

To reduce the processor count to $(n+e)/\lg n$, each processor simulates k processors per iteration (initially $k=\lg n$). After each iteration of the contraction algorithm, we balance the work among the processors by enumerating the remaining nodes in $O(\lg n)$ time via parallel prefix and compacting memory in $O(k)$ time. Thus each iterations requires $O(k \lg n)$ time. The parameter k is initially $\lg n$ and decreases exponentially with each iteration, so overall the contraction algorithm runs in $O(\lg^2 n)$ time using $(n+e)/\lg n$ processors. The balancing steps can be done without concurrent memory access, so these bounds hold for the EREW PRAM.

3 Bounded-Degree Algorithm

In this section we present a linear-processor contraction algorithm for bounded-degree graphs. We give a randomized strategy for determining a set of vertex-disjoint edges to contract on a given parallel contraction step. We discuss simplifications for the case where the graph is known to be planar or of low genus. We discuss implementation issues including maintaining linear space, using this algorithm for arbitrary graphs provided an embedding is known, making the algorithm deterministic, and reducing the processor count. Finally we mention some of the parallel models for which the analysis of section 4 holds.

The contraction algorithm for bounded-degree graphs is as follows. We assume that no vertex has degree greater than 40. The number j in line 1 is a constant chosen such that algorithm BOUNDED-CONTRACT terminates in $O(\lg n + \lg^2 \gamma)$ time with high probability. The constant will be discussed during the analysis of the algorithm in section 4.

```
Algorithm BOUNDED-CONTRACT
1 Repeat $j \lg n$ times
2   for each vertex $u$ do in parallel
3     Randomly choose an adjacent vertex $v$ such
        that MERGE($u, v$) would produce a vertex
        of degree at most 40
4   for all vertices $u$ and $v$ that choose each other,
      do MERGE($u, v$)
5   for each vertex $u$ do
      remove multiple edges to adjacent vertices
6 CONTRACT()    ;go to general contraction algorithm
```

The algorithm can be broken down into two phases: phase 1 in lines 1–5 in which vertices resulting from contraction cannot have degree greater than 40 and phase 2 in line 6 in which vertices resulting from contraction can have arbitrary degree. In section 4 we argue that phase 1 runs in $O(\lg n)$ time and with high probability reduces an n-vertex genus-γ graph to a graph whose largest connected component has $O(\gamma)$ vertices. By the argument in section 2, phase 2 reduces each $O(\gamma)$-sized connected component of a graph to a single vertex in $O(\lg^2 \gamma)$ time. Planar graphs and graphs of constant genus will be reduced to constant-sized graphs during phase 1 of algorithm BOUNDED-CONTRACT with high probability. If we know that a graph is of constant genus, we can simplify the algorithm to repeated application of lines 2–5 only, until the graph is contracted. The parameter 40 in line 3 must be replaced by $c(\gamma)$, a constant depending upon the genus γ. In particular, $c(\gamma)=40$ suffices for planar graphs. We then check for termination (all vertices degree 0) every $\lg n$ parallel contraction steps. This simplified algorithm will contract an n-vertex bounded-genus graph to a single vertex in $O(\lg n)$ time with high probabilty. Furthermore, because the degree of each vertex remains bounded throughout the algorithm in this case, we can simply assign one processor to each vertex.

The algorithm can be made to work in $O(\lg n + \lg^2 \gamma)$ time for any graph whose maximum degree is a constant greater than 40 by replacing the number 40 in line 3 with the maximum degree. Moreover, the algorithm can be made to work in $O(\lg n + \lg^2 \gamma)$ time for any graph, provided an embedding for the graph is known, since we can transform any general graph of genus γ into a degree-3 genus-γ graph by replacing each vertex by a ring of degree-3 vertices. Again, using the idea of "spares" from [20], the algorithm can be made to use only linear space.

The contraction algorithm can be modified to run in deterministic $O(\lg n \lg^* n + \lg^2 \gamma)$ time by using the $O(\lg^* n)$-time algorithm of Goldberg, Plotkin, and Shannon for 3-coloring rooted trees [11] to guarentee that a constant fraction of vertices merge on each iteration. Randomness is used only in the choice of vertex pairings in phase 1. In the deterministic version of the algorithm, each vertex u chooses the vertex v with smallest identifier such that MERGE(u, v) would produce a vertex of degree at most d_{max}, the maximum degree of any vertex. As before, edges chosen by both vertices adjacent to it are automatically selected to contract. Consider the graph induced by the edges chosen by exactly one adjacent vertex. If the edges are directed such that the vertex choosing the edge is at the tail, then this graph is a bounded-degree directed forest with edges directed from child to parent. We can then use the Goldberg-Plotkin-Shannon $O(\lg^* n)$-time algorithm for coloring bounded-degree trees to color the forest using a constant number of colors [11]. Then, we sequence through the colors allowing edges chosen by vertices of the current color to contract, provided the vertex on the other end of the edge has not already participated in a contraction during this parallel contraction step. In this scheme, we always have a constant fraction of the vertices merging.

We can reduce the processor requirement of phase 1 to $n/\lg n$ by using the techniques of Gazit and Reif [10] who in turn use the load balancing techniques of Cole and Vishkin [5]. Shannon [26] gives a scheduling algorithm that converts the deterministic $O(\lg n \lg^* n)$-time n-processor algorithm for graphs of bounded genus to an optimal $O(\lg n \lg^* n)$-time $n/(\lg n \lg^* n)$-processor algorithm.

Algorithm BOUNDED-CONTRACT is robust in that it can be implemented in several of the most restrictive parallel models. By careful attention to the data structures used to implement adjacency lists, it is possible to guarantee that no concurrent reading or writing occurs, and thus, the performance bounds apply in the EREW PRAM model. Since each processor is responsible for a single edge (or vertex) of the graph, it is naturally a "data-parallel" algorithm in the sense of [16]. Finally, the simplified version of the algorithm which uses only phase 1 is "conservative" in the sense of [20], and thus runs in $O(\lg n + \lg^2 \gamma)$ steps in the DRAM model.

4 Analysis of the Contraction Algorithm

In this section we analyze the contraction algorithm of section 3. We require two lemmas concerning bounded-degree graphs. Lemma 1 (the Missing-Edge Lemma) provides an upper bound on the number of edges in a bounded-degree graph based on Euler's formula and the number of degree-three vertices that are ineligible for contraction. The missing-edge lemma is actually proved in section 5. We use a pigeonholing argument to show in Lemma 2 that at each parallel contraction step of phase 1, a constant fraction of the vertices are adjacent to at least one edge that can be contracted. —This lemma is essentially the same as one proved independently by Boyar and Karloff [3] in the context of coloring planar graphs, but our lemma is more general and our proof differs somewhat. The final result of the section proves that with high probability, the running time of Algorithm BOUNDED-CONTRACT is $O(\lg n + \lg^2 \gamma)$.

We analyze Algorithm BOUNDED-CONTRACT using a constant $d_{\max} \geq 40$ in place of 40 in line 3 of the contraction algorithm. Using a symbolic value allows us to see in the analysis why we choose degree 40 as the maximum degree in the algorithm and to see how the contraction algorithm generalizes to bounded-degree graphs in general. In fact, a choice of $d_{\max} = 3$ suffices for Algorithm BOUNDED-CONTRACT to contract binary trees in randomized $O(\lg n)$ time.

We first present some definitions. Let $G = (V, E)$ be a graph with degree at most $d_{\max} \geq 40$. We call an edge $(u, v) \in E$ *eligible* if MERGE(u, v) would produce a vertex w of degree at most $d_{\max}$, as in line 3 of Algorithm BOUNDED-CONTRACT. Typically, the degree of w is $\deg(w) = \deg(u) + \deg(v) - 2$, but it is less whenever the adjacency lists of u and v have vertices in common because a contraction step removes multiple edges. (In fact, after the parallel contraction step implemented by Algorithm BOUNDED-CONTRACT, the degree of vertex w may be even smaller, since lines 5 and 6 remove additional multiple edges caused by other simultaneous contraction steps.) We define a vertex $u \in V$ to be *good* if it is incident on at least one eligible edge, and *bad* otherwise. A vertex of degree 1 or 2 is automatically good. A vertex v of degree 3 is good unless it is incident on three degree-$d_{\max}$ vertices v_1, v_2, and v_3 that are independent (no edge between any pair). Consequently, the bad degree-3 vertex v causes the edges (v_1, v_2), (v_2, v_3), and (v_1, v_3) to be *missing* from the graph, even though their inclusion would not increase the genus of the graph.

The next lemma uses the notion of missing edges to show that a graph with bad degree-3 vertices is sparser than required by Euler's formula alone. It is proved in section 5.

Lemma 1 *(Missing-Edge Lemma) A graph $G = (V, E)$ of genus γ_G with $b_3 \geq 0$ bad degree-3 vertices has at most $3|V| - b_3 + 10\gamma_G$ edges.*

The next lemma uses the Missing-Edge Lemma and pigeonholing to show that during each parallel paired contraction step in phase 1 of algorithm BOUNDED-CONTRACT, a constant fraction of the vertices are eligible to contract.

Lemma 2 *Any genus-γ_G graph $G = (V, E)$ with degree at most $d_{\max} \geq 40$ and $|V| \geq 40\gamma_G$ has at least $\frac{1}{12}|V|$ good vertices.*

Proof. We begin the proof by defining the following sets:

V_{good} is the set of good vertices,
V_{bad3} is the set of bad degree-3 vertices,
V_{bad456} is the set of bad vertices of degrees 4, 5, or 6,
V_{high} is the set of vertices of degrees $d_{\max} - i$,
for $i = 0, 1, 2$ or 3,
$V_{\text{rest}} = V - (V_{\text{good}} \cup V_{\text{bad3}} \cup V_{\text{bad456}} \cup V_{\text{high}})$.

having cardinalities g, b_3, b_{456}, h, and b_{rest} respectively.

We determine a lower bound on the number g of good vertices using three constraints. The first constraint is a lower bound on the number e of edges in E, which is also half the sum of the degrees of all vertices in V. For each of the sets defined above, we underestimate the sum of the degrees of the vertices in the set to yield

$$\begin{aligned} e &\geq \frac{1}{2}(g + 3b_3 + 4b_{456} + (d_{\max} - 3)h + 7b_{\text{rest}}) \\ &= \frac{1}{2}(g + 3b_3 + 4b_{456} + (d_{\max} - 3)h + \\ &\qquad 7(n - g - b_3 - b_{456} - h)). \qquad (1) \end{aligned}$$

We can use a similar technique to determine a lower bound on the number of high-degree vertices. The number of edges leaving the set V_{high} is at most $d_{\max}h$, which must be at least the number leaving the sets V_{bad3} and V_{bad456} since each of the vertices in these sets is adjacent only to vertices in V_{high}. Hence, we obtain

$$h \geq \frac{3b_3 + 4b_{456}}{d_{\max}}. \qquad (2)$$

Finally, from the Missing-Edge Lemma we have

$$e < 3n - b_3 + 10\gamma_G. \qquad (3)$$

We use these three constraints to obtain the desired lower bound on the number g of good vertices. Combining Inequalities (1) and (3) and solving for g, then substituting for h using Inequality (2) yields

$$\begin{aligned} g &> \frac{1}{6}\left[\left(\frac{d_{\max} - 30}{d_{\max}}\right)b_3 + \left(\frac{d_{\max} - 40}{d_{\max}}\right)b_{456} + n - 20\gamma_G\right] \\ &\geq \frac{1}{6}(n - 20\gamma_G), \end{aligned}$$

if $d_{\max} \geq 40$ as assumed. Since we also assume that $n \geq 40\gamma_G$, we have that $g > n/12$ which completes the proof. ■

We now analyze the behavior of phase 1. Phase 1 runs in $O(\lg n)$ time. Given a graph G with maximum vertex degree at most 40, Algorithm BOUNDED-CONTRACT does not allow the degree of any vertex to exceed 40 during phase 1 (the parallel contraction steps in lines 2–5 of Algorithm BOUNDED-CONTRACT). Therefore, each such step can be performed in constant time, since contraction and removal of multiple edges involves only communicating around constant-length vertex rings. Since we execute $j \lg n$ parallel contraction steps for a constant j, phase 1 is over in $O(\lg n)$ time.

We now use Lemma 2 to show that for suitable choice of constant j, with high probability phase 1 reduces an n-vertex genus-γ graph to a graph with $O(\gamma)$ vertices in its largest connected component. We need only show that with high probability, $O(\lg n)$ parallel contraction steps suffice to contract each connected component to size $O(\gamma)$. From lemma 2, we have that at least 1/12 of the vertices are good provided that there are at least $40\gamma_G$ vertices. The lemma applies independently to each connected component. For component i, contraction steps have a $o(1)$ probability of success only after the component size has been reduced to $O(\gamma_i)$ where γ_i is the genus of component i.

Theorem 3 *After $O(k\lg n)$ parallel paired contraction steps,any connected, degree-40, graph of genus γ has contracted to a graph with $O(\gamma)$ vertices with probability $1 - O(1/n^k)$ for any constant k.*

Proof. During each parallel contraction step, each vertex chooses an edge randomly out of all adjacent edges eligible for contraction. Since in phase 1 no vertex degree exceeds 40 throughout the contraction, on each iteration every good vertex u has at least a 1/40 probability of merging, since 1/40 is a lower bound on the probability that the edge (u, v) that vertex u chooses is also chosen by the vertex v at the other end. Lemma 2 shows that a bounded-degree genus-γ graph of size 40γ will have at least 1/12 of its vertices be good during each iteration of phase 1. Therefore, we expect that at least $n/480$ of the vertices of an n-vertex bounded-degree graph will contract on each iteration of phase 1 provided that n is sufficiently large compared to the genus of the graph. A Chernoff-bound argument completes the proof. ■

We have shown that with high probability, we require only $O(\lg n)$ parallel contraction steps to reduce connected component i to size $O(\gamma_i)$ and hence reduce the maximum connected component to size $O(\gamma)$. Therefore phase 1 runs in $O(\lg n)$ time and with high probability reduces the largest component of the graph to size $O(\gamma)$.

Let us now consider the graph at the start of phase 2. Assuming that we are not already done, we have with high probability at most $O(\gamma)$ nodes in each component. Connected subgraphs contract independently so the asymptotic contraction time during phase 2 is dominated by the time to contract the largest component. If the maximum component has size s, then by the analysis in section 2, phase 2 terminates in $O(\lg^2 s)$ time. Since we have $s = O(\gamma)$ with high probability, then phase 2 terminates in $O(\lg^2 \gamma)$ time with high probability.

Since the time to perform the contraction is simply the sum of the times to perform phase 1 and phase 2, we have that an n-vertex genus γ graph is contracted by algorithm BOUNDED-CONTRACT in $O(\lg n + \lg^2 \gamma)$ time with high probability. The probability of algorithm BOUNDED-CONTRACT failing is the probability of failing in phase 1, since phase 2 is deterministic.

The deterministic version guarantees that phase 1 reduces the n-vertex graph to an $O(\gamma)$-vertex graph, where γ is the largest genus of any connnected component. Lemma 2 guarantees that in each parallel contraction step, each n_i-vertex genus γ_i component for which $n_i = \Omega(\gamma_i)$ has at least $n_i/12$ good vertices. Of these good vertices, at least 2/41 are matched by the symmetry-breaking techniques presented in section 3. The worst case is achieved by many vertices with 40 degree-one neighbors. Therefore, each such component is reduced by at least a factor of 491/492 after each parallel contraction step. By running phase 1 of algorithm BOUNDED-CONTRACT for $\lg_{492/491} n$ iterations, we are guaranteed that each component is of size $O(\gamma_i)$ when we proceed to phase 2. In phase 1, the time required to perform each parallel contraction step is dominated by the $\Theta(\lg^* n)$ time required to color rooted trees [11]. Therefore phase 1 always terminates in $\Theta(\lg n \lg^* n)$ time. Because the graph passed to phase 2 is always of size $O(\gamma)$, phase 2 always terminates in time $O(\lg^2 \gamma)$, and therefore the deterministic contraction algorithm runs in time $O(\lg n \lg^* n + \lg^2 \gamma)$.

There is a constant-factor tradeoff between time spent in phase 1 and time spent in phase 2. The constant factor 1/12 of good nodes guaranteed by lemma 2 can be replaced by $1/c$ for any constant $c > 6$. The more general version of lemma 2 states that any bounded-degree genus-γ graph with at least $20c\gamma/(c-6)$ nodes has at least $1/c$ good nodes. If we choose to base our algorithm upon a fraction of $1/c > 1/12$ good nodes, the constant j in line 1 of algorithm BOUNDED-CONTRACT decreases. The expected size of the graph passed on to phase 2 increases, however, and therefore we can expect phase 2 to require more contraction steps.

We should comment that although the constants are large in the asymptotic bounds in Theorem 3, the analysis is highly pessimistic. Typically, a vertex has a much greater chance than 1 in 12 of being good, and if it is good, it typically has more than a 1 in 40 chance of merging with a neighbor, because the neighbor is unlikely to be incident on 40 eligible edges. Consequently, in practice we could reduce the constant j in line 1 of algorithm BOUNDED-CONTRACT without significantly harming the behavior of the contraction algorithm.

5 Missing-Edge Lemma

In this section we present the proof of the missing-edge lemma used in the analysis of algorithm BOUNDED – CONTRACT. We begin by defining *cycle splitting* of a connected genus-γ graph, a technique that will be used in the inductive proof of the missing-edge lemma. We then prove that performing cycle splitting on a genus-γ graph results in a new connected graph of genus strictly less than γ or results in two disjoint graphs whose geni sum to the original genus γ. Finally, the proof of the missing-edge lemma is a double induction on genus and number of bad degree-three vertices. Throughout this section, we assume all graphs are connected. A disconnected graph G whose genus γ_G is the sum of the geni of its connected components can only be sparser than a connected graph of genus γ_G and hence the disconnnected graph cannot have a weaker missing-edge lemma.

Suppose we have a connected graph of genus γ embedded on a surface of genus γ (e.g. a sphere with γ handles). Consider any simple cycle $\langle v_0, v_1, \dots v_{l-1}\rangle$ of the graph. As we travel along the cycle from vertex v_0 back to vertex v_0, we have a well-defined notion of right and left, since the surface upon which the graph is embedded is *orientable* [7]. Thus the vertices adjacent to each vertex v_i on the cycle can be partitioned into two sets: those that connect to the vertex from the right ($V_{i,R}$) and those that connect to the vertex from the left ($V_{i,L}$). We perform a *cycle splitting* operation as follows. We remove the cycle from the graph, split each vertex v_i on the cycle into two vertices $v_{i,R}$ and $v_{i,L}$ and form them into two cycles $\langle v_{0,R}, v_{1,R}, \dots, v_{l-1,R}\rangle$ and

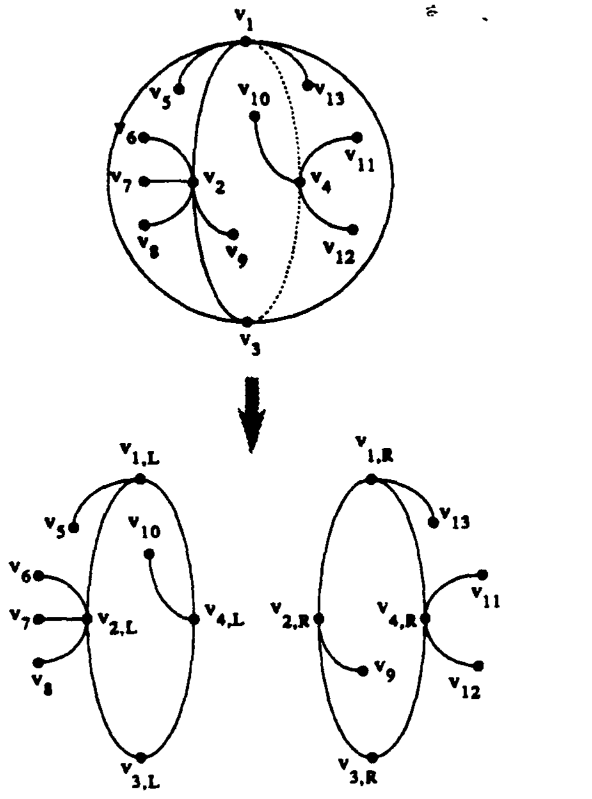

Figure 3: Given an embedding of a graph, a cycle can be split into two pieces: the right piece which is connected to nodes on the right as we tranverse the cycle, and the left piece which is connected to nodes on the left.

$\langle v_{0,L}, v_{1,L}, \ldots, v_{l-1,L}\rangle$. Finally, vertex $v_{i,R}$ is connected to each vertex in $V_{i,R}$, and vertex $v_{i,L}$ is connected to each vertex in $V_{i,L}$. Intuitively, if we view the embedded cycle as having finite thickness, we simply cut it in half. In figure 3, cycle $v_0, \ldots v_3$ is split. The left version is connected to edges entering the cycle from the left and vice versa.

The following lemma will be used to allow application of the induction hypothesis in the proof of the missing-edge lemma.

Lemma 4 *If we split a cycle $C = \langle v_0, v_1, \ldots, v_{l-1}\rangle$ of a graph G of genus γ to obtain a new graph G' of genus γ', we have either*

1. *graph G' contains two disjoint components: graph G_L of genus γ_L and graph G_R of genus γ_R such that $\gamma_L + \gamma_R = \gamma$, or*
2. *graph G' is connected and $\gamma' = \gamma - 1$.*

Proof. Suppose we have an embedding of graph G on an orientable manifold of genus γ such as a sphere with γ handles. By definition of the genus of a graph and of a surface, all the faces of graph G are simply connected [7]. That is, they can be continuously contracted to a point on the surface. Let v, e, and f be the number of vertices, edges, and faces respectively of graph G. Then the *Euler Characteristic* of the surface is defined as $\chi = v - e + f$. If

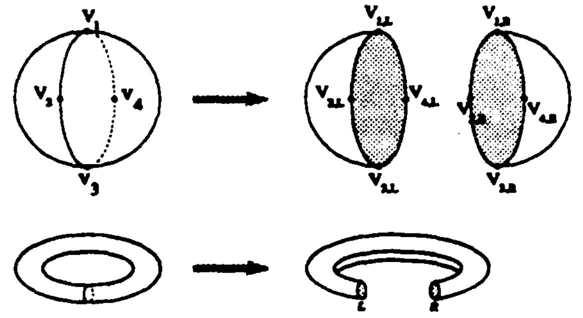

Figure 4: Given an embedding of a graph G on a surface of corresponding genus, we can remove a simple cycle of graph G from the surface and patch the resulting holes with disks. The result is an embedding of the graph G' obtained by splitting the given cycle of G. The new surface may or may not be connected.

we embed any graph on the surface such that all faces are simply connected, the quantity $v - e + f$ for that graph will always be equal to the Euler characteristic of the surface. Furthermore, by definition, we have that $\chi = 2 - 2\gamma$ where γ is the genus of the surface (and of any graph which can be embedded on that surface such that each face is simply connected).

Consider now the surface with the embedding of graph G. We cut the surface along cycle C and patch the two resulting boundaries with disks as illustrated in figure 4. The resulting surface is an orientable manifold that may or may not be connected.

Let us calculate the Euler characteristic of the new surface. *The new graph G' is embedded on the new surface* such that all faces are simply connected since the disks used to patch the cut are simply connected and no other faces of the original graph G are altered by the procedure. Therefore, if v', e', and f' are the number of vertices, edges, and faces in the cycle-split graph G', then the Euler characteristic of the new surface is equal to $\chi' = v' - e' + f' = 2 - 2\gamma'$ where γ' is the genus of graph G' and the surface upon which it is embedded. We have that $v' = v + l$ and $e' = e + l$ and $f' = f + 2$. Therefore, we have that $\chi' = \chi + 2$.

First we will consider the case where the new surface is disconnected. In this case the Euler characteristic is simply the sum of the Euler characteristics of the two pieces (the formula $v - e + f$ is additive). Let one piece have characteristic $\chi_L = 2 - 2\gamma_L$ and the other piece have characteristic $\chi_R = 2 - 2\gamma_R$. Then we have that

$$\begin{aligned} \chi' &= \chi_R + \chi_L \\ &= 2 - 2\gamma_L + 2 - 2\gamma_R \\ &= 4 - 2(\gamma_L + \gamma_R). \end{aligned}$$

We also have that $\chi' = \chi + 2 = 4 - 2\gamma$. Therefore, we have that

$$\begin{aligned} 4 - 2\gamma &= 4 - 2(\gamma_L + \gamma_R) \\ \gamma &= \gamma_L + \gamma_R \end{aligned}$$

which proves case (*1*).

Next we consider the case where the new surface is connected. Then we have that $\chi' = 2 - 2\gamma'$ and $\chi' = \chi + 2 = 4 - 2\gamma$. Therefore, equating the right-hand sides, we have that $2 - 2\gamma' = 4 - 2\gamma$ or $\gamma' = \gamma - 1$ which proves case (2). ■

Now we proceed to the proof of the missing-edge lemma. First, let us remind the reader of some terminology. Bad vertices are defined as they were in section 4: no adjacent edge can be contracted without potentially creating a vertex with degree exceeding the bound. Bad degree-three vertices must be adjacent to three degree-d_{max} independent vertices (no edges between any pair). Thus degree-three vertices force a stronger sparsity than Euler's formula alone since these three *missing* edges can be added to the graph without increasing its genus. The missing-edge lemma quantifies this increased sparsity.

For convenience we restate the lemma as it appeared in section 4, assuming this time that the graph is connected. The proof goes through for disconnected graphs where genus is defined in the usual way.

Lemma 1 (Missing-Edge Lemma) *A graph $G = (V, E)$ of genus γ with $b_3 \geq 0$ bad degree-3 vertices has at most $3|V| - b_3 + 10\gamma$ edges.*

Proof. The proof centers on showing that the number of missing edges is at least $b_3 - 4\gamma + 1$ for $b_3 > 0$, which, together with the constraint that $|E| < 3|V| + 6\gamma$ from Euler's formula, suffices to prove the lemma. The goal of the proof is to show that sharing of missing edges is limited. For the remainder of this proof, when we say "bad vertex," we assume the vertex has degree 3.

We begin our induction by showing that the lemma holds for the case $\gamma = 0$ for any number b_3 of bad vertices. That is, we show that a planar graph $G = (V, E)$ with $b_3 \geq 0$ bad degree-3 vertices has at most $3|V| - b_3$ edges.

The proof of the planar case is by induction on b_3. If $b_3 = 0$, Euler's formula alone is sufficient. The lemma holds trivially for the cases $b_3 = 1$ and $b_3 = 2$ since any graph with at least one bad degree-3 vertex has at least three distinct missing edges. Now consider the case $b_3 = 3$. If there are only three missing edges in the graph, then each of the three bad vertices is associated with the same three missing edges. Hence all three bad vertices are adjacent to the same three degree-d_{max} vertices, and hence the graph contains an instance of $K_{3,3}$, which violates the assumption that the graph is planar. Consequently, the graph has at least $4 = b_3 + 1$ missing edges.

We now consider the general planar case $b_3 \geq 4$. Assume inductively that any graph with $k < b_3$ bad vertices has at least $k + 1$ missing edges, but that there exists a planar graph $G = (V, E)$ with b_3 bad vertices and $m \leq b_3$ missing edges. Since each bad vertex is associated with exactly three missing edges, it follows that there exists a missing edge (u_1, u_2) associated with at least three bad vertices v_1, v_2, and v_3. There must be at least one additional bad vertex v_4, since we have $b_3 \geq 4$. For some embedding of G on the sphere, the Jordan Curve Theorem ensures that two of the vertices, say v_1 and v_2, together with u_1 and u_2 form a cycle C that separates v_3 from v_4, as shown in Figure 5.

We are now set up to apply the induction hypothesis. Let G_{in} be the subgraph of G induced by cycle C and the vertices on one side of C, and let G_{out} be the subgraph induced by cycle C and the vertices on the other side of C. Add to u_1 and u_2 in each of G_{in} and G_{out} enough degree-1 neighbors to maintain their degrees as d_{max}. Let b_{in} be the number of bad (degree-3) vertices in G_{in}, and let b_{out} be the number of bad vertices in G_{out}. Since no new bad degree-3 vertices are introduced into G_{in} and G_{out}, we have $b_{in} < b_3$ and $b_{out} < b_3$. By the induction hypothesis, therefore, graph G_{in} has at least $b_{in} + 1$ missing edges, and

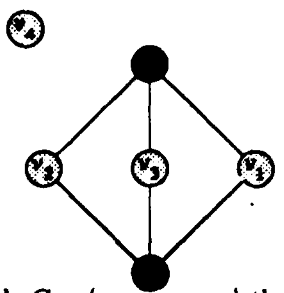

Figure 5: A cycle $C = \langle u_1, v_1, u_2, v_2 \rangle$ that separates G into two subgraphs G_{in} and G_{out}, each with at most $b_3 - 1$ bad degree-3 vertices.

graph G_{out} has at least $b_{out} + 1$ missing edges.

We next account for interactions between graphs G_{in} and G_{out} to obtain the lower bound of $b_3 + 1$ on the number of missing edges in the original graph G. First, observe that $b_{in} + b_{out} = b_3$, since when v_1 is a bad vertex in one of the graphs, it is a degree-2 vertex in the other, and similarly for v_2. Moreover, each of v_1 and v_2 are bad in at least one of G_{in} and G_{out} because we dummied up each of u_1 and u_2 to have degree d_{max}. The number of missing edges in G is at least $(b_{in} + 1) + (b_{out} + 1)$ minus the number of missing edges shared by G_{in} and G_{out}. The number of such shared missing edges is in fact 1, namely, the edge (u_1, u_2), since u_1 and u_2 are the only degree-d_{max} vertices in both G_{in} and G_{out}. Consequently, the number of missing edges in G is at least $(b_{in} + 1) + (b_{out} + 1) - 1 = b_3 + 1$, which completes proof of the planar version of the lemma.

We have shown that the missing edge lemma lemma holds for the case $\gamma = 0$ for any number b_3 of bad vertices. To complete the base cases, we see that for general graphs the lemma holds for $b_3 = 0$ trivially using only Euler's formula. It also holds trivially for $1 \leq b_3 < 4\gamma + 3$ for all γ since then the lemma only requires that the number of missing edges is at least 3 which is true for any graph with at least one bad vertex.

We now consider the general case $b_3 \geq 4\gamma + 3$ and $\gamma \geq 1$. We assume inductively that any genus-γ graph with $k < b_3$ bad vertices has at least $k - 4\gamma + 1$ *missing* edges and any graph with genus $j < \gamma$ and any number b_3 bad degree-three nodes has at least $b_3 - 4j + 1$ missing edges. Assume, however that there exists a graph $G = (V, E)$ of genus γ with $b_3 \geq 4\gamma + 4$ bad vertices and $m \leq b_3 - 4\gamma$ missing edges. Since each bad vertex is associated with exactly three missing edges, it follows that there exists a missing edge (u_1, u_2) associated with at least three bad vertices v_1, v_2, and v_3. There must be at least one additional bad vertex v_4 since we have $b_3 \geq 7$. The situation is illustrated in figure 5.

To allow application of the induction hypothesis, we split graph G along one of the three simple cycles shown in figure 5 to obtain a new graph G' of genus γ'. As we did in proving the planar version of this lemma, we add degree-one neighbors to the high-degree nodes u_1 and u_2 so that each of the four nodes resulting from the split of u_1 and u_2 has maximum degree. Each bad degree-three node on the cycle is split into two nodes, but exactly one of these two nodes is a bad degree-three node in the new graph G'. The other node split from that vertex is of degree 2. Therefore graph G', whether connected or not, has exactly b_3 bad nodes.

Suppose we can split graph G along one of the cycles such that the resulting graph G' is still connected. Then by lemma 4(*2*) we have that $\gamma' = \gamma - 1$. As argued above, graph G' has the same number b_3 of bad degree-three vertices as the original graph did. Because graph G' has lower genus, we can apply the induction hypothesis. Therefore, graph G' has at least $b_3 - 4(\gamma - 1) + 1 = b_3 - 4\gamma + 5$ missing edges. We overcounted exactly one missing edge, namely (u_1, u_2), since these are the only two degree-d_{max} vertices duplicated. Therefore we have at least $b_3 - 4\gamma + 4 > b_3 - 4\gamma + 1$ missing edges.

Now suppose that splitting along any of the three simple cycles separates G' into two graphs: graph G_L with genus γ_L and b_L bad degree-three vertices, and graph G_R with genus γ_R and b_R bad degree-three vertices. Suppose that splitting along one of the cycles yields graphs such that $\gamma_L < \gamma$ and $\gamma_R < \gamma$. Applying the induction hypothesis to each side we have that the number of missing edges in graph G_R is at least $b_R - 4\gamma_R + 1$ and the number of missing edges in graph G_L is at least $b_L - 4\gamma_L + 1$. Adding the missing edges together and subtracting one for the overcount of edge (u_1, u_2), we have that the original graph G has at least $(b_L + b_R) - 4(\gamma_L + \gamma_R) + 2 - 1$ missing edges. From lemma 4(*1*) we have that $\gamma_L + \gamma_R = \gamma$ and we argued earlier that $b_L + b_R = b_3$. Substituting these back into the expression for the number of missing edges in graph G, we have that graph G has at least $b_3 - 4\gamma + 1$ missing edges.

The final case we must consider is that each of the three cycles cuts off a planar patch. In this case we have a situation analogous to the planar case argued earlier. As illustrated in figure 5, one of the cycles cuts the graph into two graphs G_L and G_R such that $b_L < b_3$ and $b_R < b_3$. Without loss of generality, let us assume that graph G_R is planar. Then by lemma 4(*1*), graph G_L must have genus γ. Applying the induction hypothesis to the two graphs (since each has less than b_3 bad vertices), we have that G_R has at least $b_R + 1$ missing edges and G_L has at least $b_L - 4\gamma + 1$ missing edges. Again we have that $b_R + b_L = b_3$. Adding the missing edges and subtracting one for the overcount of edge (u_1, u_2), we have that graph G has at least $b_L + b_R - 4\gamma + 2 - 1 = b_3 - 4\gamma + 1$ missing edges which completes the proof. ∎

6 Applications

The most direct application of the parallel contraction algorithm is to the vision problem of determining the connected regions of an image represented as a two-dimensional array of pixels. We can view the image as a planar graph $G = (V, E)$, where the vertex set V is the set of pixels, and $(u, v) \in E$ if u and v are adjacent pixels with the same color. The *region labeling* problem is to assign each pixel an integer such that two pixels have the same label if and only if they are path connected in G.

Region labeling can be solved by a connected-components algorithm. Shiloach and Vishkin [27] give an $O(\lg n)$-time, n-processor parallel algorithm for connected components using the concurrent-read, concurrent-write (CRCW) PRAM model. Hagerup [13] gives an $O(\lg n)$-time, $n/\lg n$-processor CRCW algorithm for graphs of bounded genus[1]. The best algorithm to date in the more easily implemented exclusive-read, exclusive-write (EREW) model is due to Hirschberg, Chandra, and Sarwate [17], who give an $O(\lg^2 n)$ time algorithm for connected components using the adjacency-matrix representation of a graph. Leiserson and Maggs [20] give an $O(\lg^2 n)$ step, n-processor, randomized connected-components algorithm for a DRAM (distributed random-access machine), an EREW-like model that includes the cost of communication. Blelloch [2] gives an $O(\lg n)$ randomized algorithm for a model that includes parallel prefix as a primitive operation. Lim [21] gives a region-labeling algorithm that runs in $O(\lg^2 n)$ time on an EREW PRAM which uses the planar and geometric properties of a mesh. Gazit and Reif have recently developed a randomized $O((n+m)/\lg n)$-processor EREW algorithm that runs in time $O(\lg n + \lg^2 g)$ where g is the genus of the graph (what we call γ_G, which can be much larger than γ in disconnected graphs). They require an embedding of the graph. Other algorithms for connected components are given by [6, 9, 19, 24, 30, 32].

Our algorithm for labeling planar graphs is asymptotically the fastest algorithm to date in the EREW PRAM model. The algorithm uses Algorithm BOUNDED-CONTRACT to simultaneously contract each component of the graph to a single node, and then it simply reverses the contraction process and assigns the same label to all nodes in each connected component. The algorithm uses $O(n/\lg n)$processors, and with high probability (*i.e.*, probability at least $1 - O(1/n^k)$ for any constant k), the algorithm runs in $O(\lg n)$ time on an EREW PRAM or a DRAM. Thus, our algorithm achieves the best possible asymptotic bound in the most restrictive parallel models. The deterministic version of the algorithm runs in $O(\lg n \lg^* n)$ time using an optimal number of processors $(n/\lg n \lg^* n)$. Like Lim's $O(\lg^2 n)$-time algorithm, our algorithm for region labeling takes advantage of the planar nature of the image graph, but unlike his algorithm, it does not depend on the geometric nature of the image.[2] Thus, our algorithm works on irregular planar structures, including meshes with local refinements and meshes with (static) faulty elements.

The contraction algorithm can also be used for a variety of other applications including algorithms for biconnected components of planar graphs and for spanning trees of planar graphs. The running times for these algorithms is asymptotically the same as for the contraction algorithm, and for these problems the running times are asymptotically the best to date in the EREW model. All the algorithms, including the connected components algorithm, can be generalized to graphs of higher genus and higher degree using algorithm BOUNDED-CONTRACT. The asymptotic running times for these problems match the running times of the contraction algorithm (i. e. $O(\lg n + \lg^2 \gamma)$ with high probability where γ is the maximum genus of any connected component). For classes of graphs with genus $o(n)$ this is the best bound to date. For graphs of genus $\Omega(n)$ this matches the best previous bound of $O(\lg^2 n)$.

[1]Hagerup has recently independently developed an $O(\lg n \lg^* n)$-time $n/\lg n \lg^* n$-processor deterministic EREW algorithm for graphs of bounded genus [14]

[2]Subsequent to the development of the planar version of our algorithm, Lim, Agrawal, and Nekludova [22] obtained an $O(\lg n)$-time deterministic algorithm for region labeling. Like Lim's earlier work, their algorithm exploits the geometric properties of specific planar grids.

7 Conclusion

In this section we comment upon the practical behavior of algorithm BOUNDED-CONTRACTand present some open problems.

In section 4 we commented that the behavior of algorithm BOUNDED-CONTRACTin practice is likely to be much better than that guarenteed by our worst-case analysis. This is particularly true for the bounded-genus case where we explicitly check for termination. For example, we are not likely to require the worst-case number of iterations to reduce each component of a planar graph to a single node. The general algorithm, however, always performs the worst-case number of iterations for phase 1 because we do not have a means of detecting when phase 1 is completed unless we in fact complete the entire contraction. Phase 2 only performs as many contraction steps as necessary. Because of this control structure, it is probably good in practice to choose the constant j in line 1 of algorithm CONTRACT based upon lemma 2 with a fraction of $1/c$ good nodes for constant $c = 6 + \epsilon$ and $\epsilon > 0$.

To detect the termination of phase 1 we require an $O(\lg n)$ time n processor algorithm which can determine the genus of an n node graph to within a constant without knowing the value of n. The problem appears difficult since determining the exact genus γ of an n-node graph is NP-complete [29] and the best sequential algorithm for this problem runs in $O(n^{O(\gamma)})$ time [8]. There seems to be no work on parallel algorithms which approximate the genus of a graph.

Acknowledgements

Thanks to Charles Leiserson of MIT who helped extensively with the development of the contraction algorithm for bounded-degree graphs and provided expert commentary on early drafts of this paper. Thanks also to Gary Miller of USC and Miller Maley of Princeton for general assistance on topology issues, to Washington Taylor of Thinking Machines Corporation who provided the key topological ideas in the proof of lemma 4, and to James Park and Tom Cormen of MIT for their help with the figures and text formatting.

References

[1] T. O. Binford, "Survey of model-based image analysis systems," *The International Journal of Robotics Research,*, Vol. 1, No. 1, 1982, pp. 18–64.

[2] G. Blelloch, "Parallel prefix vs. concurrent memory access," Technical Report, Thinking Machines Corporation, 1986.

[3] J. Boyar and H. Karloff, "Coloring planar graphs in parallel," *Journal of Algorithms*, Vol. 8, 1987, pp. 470–479.

[4] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *Annals of Mathematic Statistics*, Vol. 23, 1952, pp. 493–507.

[5] R. Cole and U. Vishkin, "Approximate and exact parallel scheduling with applications to list, tree, and graph problems," *Proceedings of the 27th Annual IEEE Symposium on Foundation of Computer Science*, Oct. 1986, pp. 478–491.

[6] F. Chin, J. Lam, and I. Chen, "Efficient parallel algorithms for some graph problems," *Communications of the ACM*, Vol. 25, No. 9, September 1982, pp. 659–665.

[7] H. Graham Flegg, *From Geometry to Topology*, The English Universities Press Ltd., distributed in the United States by Crane, Russak and Company, New York, NY, 1974.

[8] I. S. Filotti, F. L. Miller, and J. Reif, "On determining the genus of a graph in $O(v^{O(g)})$ steps," *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, 1979, pp. 27–37.

[9] H. Gazit, "An optimal randomized parallel algorithm for finding connected components in a graph," *Proceedings of the 27th Annual IEEE Symposium on the Foundations of Computer Science*, 1986, pp. 492–501.

[10] H. Gazit and J. Reif "A randomized EREW parallel algorithm for finding the connected components in a graph," unpublished manuscript, 1988.

[11] A. V. Goldberg, S. A. Plotkin, and G. Shannon, "Parallel symmetry breaking in sparse graphs," *SIAM Journal of Discrete Math*, to appear.

[12] M. J. Greenberg and J. R. Harper, *Algebraic Topology: A First Course*, Addison-Wesley, New York, NY, 1981.

[13] T. Hagerup, "Optimal parallel algorithms for planar graphs," *Proceedings of AWOC*, 1988.

[14] T. Hagerup, "Optimal parallel algorithms for planar graphs," *Information and Computation*, to appear.

[15] F. Harary, *Graph Theory*, Addison-Wesley, Reading, MA, 1962.

[16] W. D. Hillis and G. L. Steele, Jr., "Data parallel algorithms," *Communications of the ACM*, Vol. 29, No. 12, December 1986, pp. 1170–1183.

[17] D. S. Hirshberg, A. K. Chandra, and D. V. Sarwate, "Computing connected components on parallel computers," *Communications of the ACM*, Vol. 22, No. 8, August 1979, pp. 461–464.

[18] B. K. P. Horn, *Robot Vision*, MIT Press, Cambridge, MA, 1986.

[19] V. Koubek and J. Kršňáková, "Parallel algorithms for connected components in a graph," *Proceedings of the 5th International Conference on Fundamentals of Computation Theory*, Springer Lecture Notes in Computer Science, Vol. 199, pp. 208–217.

[20] C. E. Leiserson and B. M. Maggs, "Communication-efficient parallel algorithms for distributed random-access machines," *Algorithmica* Vol. 3, No. 1, 1988 pp. 53–78.

[21] W. Lim, "Fast algorithms for labeling connected components in 2-D arrays," unpublished manuscript, MIT, July 1986.

[22] W. Lim, A. Agrawal, and L. Nekludova, "A parallel $O(\lg n)$ algorithm for finding connected components in planar images," *Proceedings of the 1987 International Conference on Parallel Processing*, August 1987, pp. 783–786.

[23] G. Miller and J. Reif, "Parallel tree contraction and its application," *Proceedings of the 26th Annual IEEE Symposium on the Foundations of Computer Science*, 1985, pp. 478–489.

[24] D. Nath and S. N. Maheshwari, "Parallel algorithms for the connected components and minimal spanning tree problems," *Information Processing Letters*, Vol. 14, pp. 7–11.

[25] C. Savage and J. JáJá, "Fast, efficient parallel algorithms for some graph problems," *SIAM Journal of Computing*. Vol. 10, No. 4, November 1981, pp. 682–691.

[26] G. E. Shannon, "Reassigning tasks to processors less frequently: a technique for designing parallel algorithms," unpublished manuscript, Purdue University, October 1987.

[27] Y. Shiloach and U. Vishkin, "An $O(\lg n)$ parallel connectivity algorithm," *Journal of Algorithms*, Vol. 3, 1982, pp. 57–67.

[28] R. E. Tarjan and U. Vishkin, "Finding biconnected components and computing tree functions in logarithmic parallel time," *Proceedings of the 24th Annual IEEE Symposium on the Foundations of Computer Science*, 1984, pp. 12–20.

[29] C. Thomassen, "The graph genus problem is NP-complete", *J. Algorithms*, to appear.

[30] U. Vishkin, "An optimal parallel connectivity algorithm," RC5149, IBM T. J. Watson Research Center, Yorktown Heights, NY, 1981.

[31] U. Vishkin, "Synchronous parallel computation—a survey," unpublished manuscript, Courant Institute, New York University, April 1983.

[32] J. C. Wyllie, "The complexity of parallel computation, TR 79-387, Dept. of Computer Science, Cornell University, Ithaca, NY, 1979.